

---

# **django-allauth-2fa Documentation**

***Release 0.4.3***

**Víðir Valberg Guðmundsson, Percipient Networks**

**May 05, 2021**



---

## Contents:

---

<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>Compatibility</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>7</b>
3.1	Running tests . . . . .	7
3.2	Running the test project . . . . .	7
<b>4</b>	<b>Indices and tables</b>	<b>13</b>



django-allauth-2fa adds [two-factor authentication](#) to [django-allauth](#). [django-allauth](#) is a set of [Django](#) applications which help with authentication, registration, and other account management tasks.

**Source code** <http://github.com/percipient/django-allauth-2fa>

**Documentation** <https://django-allauth-2fa.readthedocs.io/>



# CHAPTER 1

---

## Features

---

- Adds [two-factor authentication](#) views and workflow to [django-allauth](#).
- Supports Authenticator apps via a QR code when enabling 2FA.
- Supports single-use back-up codes.





## CHAPTER 2

---

### Compatibility

---

django-allauth-2fa attempts to maintain compatibility with supported versions of Django, django-allauth, and django-otp. Current minimum versions are listed below:

- Django 1.11
- django-allauth 0.25.0
- django-otp 0.3.12
- Python 3.5



django-allauth-2fa was initially created by [Víðir Valberg Guðmundsson \(@valberg\)](#), and is currently maintained by [Percipient Networks](#). Please feel free to contribute if you find django-allauth-2fa useful!

1. Check for open issues or open a fresh issue to start a discussion around a feature idea or a bug.
2. If you feel uncomfortable or uncertain about an issue or your changes, feel free to email [support@percipientnetworks.com](mailto:support@percipientnetworks.com) and we will happily help you.
3. Fork [the repository](#) on GitHub to start making your changes to the **master** branch (or branch off of it).
4. Write a test which shows that the bug was fixed or that the feature works as expected.
5. Send a pull request and bug the maintainer until it gets merged and published.

### 3.1 Running tests

Tests can be run using the standard Django testing facility:

```
python manage.py test
```

### 3.2 Running the test project

The test project can also be used as a minimal example using the following:

```
# Migrate the SQLite database first.
DJANGO_SETTINGS_MODULE=tests.run_settings python manage.py migrate
# Run the server with debug.
DJANGO_SETTINGS_MODULE=tests.run_settings python manage.py runserver_plus
# Run the shell.
DJANGO_SETTINGS_MODULE=tests.run_settings python manage.py shell_plus
```

### 3.2.1 Installation

Install *django-allauth-2fa* with pip (note that this will install Django, django-allauth, django-otp, qrcode and all of their requirements):

```
pip install django-allauth-2fa
```

After all the pre-requisites are installed, django-allauth and django-otp must be configured in your Django settings file. (Please check the [django-allauth documentation](#) and [django-otp documentation](#) for more in-depth steps on their configuration.)

```
INSTALLED_APPS = (
    # Required by allauth.
    'django.contrib.sites',

    # Configure Django auth package.
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',

    # Enable allauth.
    'allauth',
    'allauth.account',

    # Configure the django-otp package.
    'django_otp',
    'django_otp.plugins.otp_totp',
    'django_otp.plugins.otp_static',

    # Enable two-factor auth.
    'allauth_2fa',
)

MIDDLEWARE_CLASSES = (
    # Configure Django auth package.
    'django.contrib.auth.middleware.AuthenticationMiddleware',

    # Configure the django-otp package. Note this must be after the
    # AuthenticationMiddleware.
    'django_otp.middleware.OTPMiddleware',

    # Reset login flow middleware. If this middleware is included, the login
    # flow is reset if another page is loaded between login and successfully
    # entering two-factor credentials.
    'allauth_2fa.middleware.AllauthTwoFactorMiddleware',
)

# Set the allauth adapter to be the 2FA adapter.
ACCOUNT_ADAPTER = 'allauth_2fa.adapter.OTPAdapter'

# Configure your default site. See
# https://docs.djangoproject.com/en/dev/ref/settings/#sites.
SITE_ID = 1
```

After the above is configure, you must run migrations.

```
python manage.py migrate
```

Finally, you must include the django-allauth-2fa URLs:

```
from django.conf.urls import include, url

urlpatterns = [
    # Include the allauth and 2FA urls from their respective packages.
    url(r'^$', include('allauth_2fa.urls')),
    url(r'^', include('allauth.urls')),
]
```

**Warning:** Any login view that is *not* provided by django-allauth will bypass the allauth workflow (including two-factor authentication). The Django admin site includes an additional login view (usually available at `/admin/login`).

The easiest way to fix this is to wrap it in `login_required` decorator (the code only works if you use the standard admin site, if you have a custom admin site you'll need to customize this more):

```
from django.contrib import admin
from django.contrib.auth.decorators import login_required

# Ensure users go through the allauth workflow when logging into admin.
admin.site.login = login_required(admin.site.login)
# Run the standard admin set-up.
admin.autodiscover()
```

## 3.2.2 Configuration

### ALLAUTH\_2FA\_TEMPLATE\_EXTENSION

Allows you to override the extension for the templates used by the internal views of django-allauth-2fa.

Defaults to `ACCOUNT_TEMPLATE_EXTENSION` from allauth, which is `html` by default.

This can be used to allow a different template engine for 2FA views.

### ALLAUTH\_2FA\_ALWAYS\_REVEAL\_BACKUP\_TOKENS

Whether to always show the remaining backup tokens on the Backup Tokens view, or only when they're freshly generated.

Defaults to `True`.

## 3.2.3 Advanced Configuration

### Forcing a User to Use 2FA

A User can be forced to use 2FA based on any requirements (e.g. superusers or being in a particular group). This is implemented by subclassing the `allauth_2fa.middleware.BaseRequire2FAMiddleware` and implementing the `require_2fa` method on it. This middleware needs to be added to your `MIDDLEWARE_CLASSES` setting.

For example, to require a user to be a superuser:

```
from allauth_2fa.middleware import BaseRequire2FAMiddleware

class RequireSuperuser2FAMiddleware(BaseRequire2FAMiddleware):
    def require_2fa(self, request):
        # Superusers are require to have 2FA.
        return request.user.is_superuser
```

If the user doesn't have 2FA enabled they will be redirected to the 2FA configuration page and will not be allowed to access (most) other pages.

## 3.2.4 Changelog

### 0.8 February 3, 2020

- Drop support for Python 2.7 and Python 3.4.
- Officially support Python 3.7 and 3.8.
- Drop support for Django 2.0 and Django 2.1.
- Officially support Django 3.0.

### 0.7 September 10, 2019

- Remove more code that was for Django < 1.11.
- Officially support Django 2.0 and Django 2.1.
- Officially support django-otp 0.7.
- Do not include test code in distribution, fix from @akx, PR #67.
- Support for more complex user IDs (e.g. UUIDs), fix from @chromakey, see issue #64 / PR #66.
- The extension used by the 2FA templates is customizable. Originally in PR #69 by @akx, split into PR #71.
- The QR code is now included inline as an SVG instead of being a separate view. PR #74 by @akx.
- A new mixin is included to enforce a user having 2FA enabled for particular views. Added in PR #73 by @akx.
- Passing additional context to the `TwoFactorBackupTokens` was broken. This was fixed in PR #73 by @akx.
- A configuration option (`ALLAUTH_2FA_ALWAYS_REVEAL_BACKUP_TOKENS`) was added to only show the static tokens once (during creation)> PR #75 by @akx.

### 0.6 February 13, 2018

- Drop support for Django < 1.11, these are no longer supported by django-allauth (as of 0.35.0).

### 0.5 December 21, 2017

- Avoid an exception if a user without any configured devices tries to view a QR code. This view now properly 404s.
- Redirect users to configure 2FA if they attempt to configure backup tokens without enabling 2FA first.
- Add base middleware to ensure particular users (e.g. superusers) have 2FA enabled.

- Drop official support for Django 1.9 and 1.10, they're [no longer supported](#) by the Django project.
- Added Sphinx-generated documentation. A rendered version [is available at](#).

#### 0.4.4 March 24, 2017

- Adds trailing slashes to the URL patterns. This is backwards compatible with the old URLs.
- Properly support installing in Python 3 via PyPI.

#### 0.4.3 January 18, 2017

- Adds support for forwarding GET parameters through the 2FA workflow. This fixes `next` not working when logging in using 2FA.

#### 0.4.2 December 15, 2016

- Reverts the fix in 0.4.1 as this breaks custom adapters that inherit from `OTPAdapter` and *don't* override the `login` method.

#### 0.4.1 December 14, 2016

- Fixed a bug when using a custom adapter that doesn't inherit from `OTPAdapter` and that overrides the `login` method.

#### 0.4 November 7, 2016

- Properly continue the allauth login workflow after successful 2FA login, e.g. send allauth signals
- Support using `MIDDLEWARE` setting with Django 1.10.
- Support customer `USERNAME_FIELD` on the auth model.

#### 0.3.2 October 26, 2016

- Fix an error when hitting the `TwoFactorBackupTokens` view as a non-anonymous user.

#### 0.3.1 October 5, 2016

- Properly handle an `AnonymousUser` hitting the views.

#### 0.3 October 5, 2016

- Support custom `User` models.
- Fixed a bug where a user could end up half logged in if they didn't complete the two-factor login flow. A user's login flow will now be reset. Requires enabled the included middle: `allauth_2fa.middleware.AllauthTwoFactorMiddleware`.
- Disable autocomplete on the two-factor code input form.
- Properly redirect anonymous users.

- Minor simplifications of code (and inherit more code from django-otp).
- Minor updates to documentation.

## **0.2 September 9, 2016**

- Add tests / tox / Travis support.
- Don't pin dependencies.
- Officially support Django 1.10, drop support for Django 1.7.

### **0.1.4 May 2, 2016**

- Autofocus the token input field on forms.

### **0.1.3 January 20, 2016**

- Fix deprecation notices for Django 1.10.

### **0.1.2 November 23, 2015**

- Fixed an error when a user enters invalid input into the token form.

### **0.1.1 October 21, 2015**

- Project reorganization and clean-up.
- Added support for Microsoft Authenticator.
- Support being installed via pip.
- Pull more configuration from Django settings (success URL).
- Support disabling two-factor for an account.

## **0.1 April 4, 2015**

- Initial version by Víðir Valberg Guðmundsson



## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`